**Final Action Deficient Because It Does Not Address Applicant's Arguments (Retrieving XML Syntax Info. for Routing)**

Applicant stressed on pages 13-16 of the August 16, 2005 Amendment that the applied references neither disclosed nor suggested the claimed feature of the ***router*** "initiating selected application operations for ***routing the received message based on interpreting the prescribed attributes from the XML tags*** ... [including] ***retrieving the prescribed syntax and semantics information from a selected one of a plurality of vocabulary modules***...."

In particular, Applicant stressed that the proposed modification of Abjanic to include the teachings of Shafer did not disclose or suggest the claimed invention because Shafer consistently taught that the disclosed render library (cited by the Examiner as teaching the claimed vocabulary library) was executed ***in the client device, and not in the router, as claimed.***

Applicant argued explicitly that:

> Shafer does not teach or suggest retrieving syntax and semantics information from a vocabulary library (render library) for routing the received message to the destination node. Shafer in fact teaches that the render library is in the client, not in the router.
>
> (Page 14, paragraph 2, emphasis in original).
>
> Thus, Shafer teaches that a network administrator can use the render library 70 to configure the router. The router in Shafer does not utilize the render library 70 during routing to understand routing attributes in XML language. Shafer specifically teaches that the render library 70 "renders the extracted information to a human-readable format... ." (See paragraph 50 of Shafer).
>
> In contrast, each of the independent claims specify that the router accesses the XML library to interpret the parsed XML tag from the received message to determine where the message is routed.
>
> (Page 14, paragraphs 3-4, emphasis in original)

Applicant also presented arguments on page 16 that the hypothetical combination would neither disclose nor suggest the claimed feature of interpreting the XML tags for routing based on retrieving information from a vocabulary module, because the hypothetical combination assumes

a static configuration, hence there was no suggestion of any motivation for providing flexibility in interpreting different XML-based routing protocols.

*None* of these arguments have been addressed in the Final Action. Rather, the Final Action ignores these arguments with the sweeping mischaracterization that "applicant argues that Shafer fails to disclose retrieving syntax and semantic information from a vocabulary library (page 14)." This simplistic assertion is per se unreasonable because it fails to address the ***entire purpose of retrieving the syntax information, namely routing the packet***.

Hence, the rejection still is legally deficient because it fails to address each and every element of the claim, namely retrieving the syntax information to interpret the XML tags in the packet for routing the packet. As specified in MPEP §2143.03, entitled "**All Claim Limitations Must Be Taught or Suggested**": "To establish prima facie obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). 'All words in a claim must be considered in judging the patentability of that claim against the prior art.' *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970)." MPEP §2143.03 at 2100-139 (Rev. 3, Aug. 2005).

**Applied References Fail to Teach *Interpreting* XML Tags For Routing**

The Examiner's misplaced reference to Column 11, lines 48-61 of Abjanic on page 9 demonstrates that Abjanic does not, in fact, ***interpret*** the received XML tags, because Abjanic recognizes that ***it is impossible for the director 145 to be aware of each and every XML language!*** Abjanic accepts the limitation that its director 145 in the traffic manager 140 will *not* be able to interpret all XML tags. (See Fig. 5, and col. 11, lines 22 to col 12, line 10).

Specifically, Abjanic recognizes that there are numerous XML languages that are mutually incompatible:

> As noted above, XML does not define a fixed set of tags, but rather, only defines a syntax or structured format through which users can define their own set of tags or their own XML based language. In fact there are many different XML-based languages in use, each

having a unique set of tags that define what elements should be provided to comply with that XML language.

(Col. 11, lines 25-32).

Abjanic therefore recognizes that XML messages can be routed, regardless of the XML-based language used by the message, by avoiding interpreting the XML message and instead relying on pattern matching to detect prescribed data within the XML data:

> According to an advantageous aspect of the present invention, director 145 can receive an XML message, ***compare the application data or business transaction information to the configuration pattern***, and then direct or route the message (or make switching or routing decisions) to an appropriate processing node or server ***regardless of the type of XML-based language used by the message***. Once the director 145 is configured to ***detect or recognize one or more specific tags and corresponding data*** (e.g., PurchaseAmount >$100), the director 145 can direct or route the message based on the content of the application data (e.g., based on the business transaction information provided as XML data), regardless of the type of XML-based language that is used by the message.

Hence, Abjanic relies on configuration patterns to ***detect or recognize a specific tag and corresponding data***; the configuration patterns also include routing information that enables the message director 145 to forward the packet to the appropriate XML Server (e.g., 510, 515, or 520 of Fig. 5) that is capable of interpreting the specific XML language (see col. 11, line 62 to col. 12, line 28). A detailed description of the configuration patterns is found at e.g., col. 6, line 50 to col. 8, line 4, which demonstrate that the director 145 determines whether the XML data matches a specific XML *pattern*..

Hence, Abjanic performs parsing of XML tags in order to identify selected portions for pattern matching and then performs a pattern-based analysis to determine if the selected portions match any of the configuration patterns.

In contrast, each of the independent claims specify ***interpreting*** the prescribed attributes from the XML tags by the router. The claimed term "interpreting" is interpreted in view of the explicit claim language that specifies the interpreting of the prescribed attributes from the XML tags is based on ***runtime execution of the application resource***. As described in the

specification at page 1, lines 16-18 that "the interpretation of that attribute (i.e., the attribute's 'context') is determined by the application runtime environment of the corresponding application under execution." In other words, the interpreting cannot be performed *unless* the ***context*** for the XML tags has been established.

The specification also describes that "the runtime execution of the application resource provides application-specific syntax and semantics enabling interpretation of the parsed XML tags. ... Hence, the ***router is able to determine context*** for XML tags within an XML portion of a received message in order to select application operation to be performed, ***based on the context established during runtime execution of the application resource***." (Page 3, lines 9-16).

In fact, the claims explicitly require that the interpreting of the prescribed attributes from the XML tags is based on ***runtime execution of the application resource***.

As apparent from the above-quoted description of Abjanic at col. 11, lines 49-61, Abjanic cannot perform interpreting because it performs a pattern matching of the XML data relative to a "configuration pattern" in order to "route the message based on the ***content*** of the application data ... ***regardless of the type of XML-based language that is used by the message***." (Col. 11, lines 58-61). Abjanic expressly avoids interpreting the XML tags for the very reason that "there are many different XML-based languages in use, each having a unique set of tags that define what elements should be provided to comply with that XML language" (col. 11, lines 29-32).

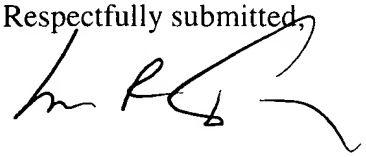
Hence, not only does Abjanic fail to disclose the claimed "interpreting" the XML tags, but also teaches away from interpreting the XML tags by relying on pattern matching relative to configuration pattersn in order to route messages "***regardless of the type of XML-based language that is used by the message***" (col. 11, lines 60-61). Consequently, one skilled in the art would avoid adding ***any*** additional vocabulary modules for interpreting XML languages because they would be unnecessary: Abjanic performs pattern matching, and expressly avoids interpreting XML tags.

For these and other reasons, the outstanding §103 rejection should be withdrawn.

In view of the above, it is believed this application is in condition for allowance, and such a Notice is respectfully solicited.

To the extent necessary, Applicant petitions for an extension of time under 37 C.F.R. 1.136. Please charge any shortage in fees due in connection with the filing of this paper, including any missing or insufficient fees under 37 C.F.R. 1.17(a), to Deposit Account No. 50-1130, under Order No. 95-457, and please credit any excess fees to such deposit account.

Respectfully submitted,



Leon R. Turkevich
Registration No. 34,035

Customer No. 23164
(202) 261-1059
**Date: December 20, 2005**